CEWES MSRC/PET TR/98-43

Microsoft DirectPlay meets DMSO RTI for Virtual Prototyping in HPC T&E Environments

by

G.C. Fox
W. Furmanski
S. Nair
Z. Odcikin Ozdemir

DoD HPC Modernization Program
Programming Environment and Training
CEWES MSRC

08h00198

Work funded wholly or in part by the DoD High Performance Computing Modernization Program CEWES Major Shared Resource Center through

Programming Environment and Training (PET)

Supported by Contract Number: DAHC 94-96-C0002
Nichols Research

Microsoft DirectPlay meets DMSO RTI for Virtual Prototyping in HPC T&E Environments

G.C. Fox, W. Furmanski, S. Nair and Z. Odcikin Ozdemir

Northeast Parallel Architectures Center, Syracuse University, Syracuse NY 13244-4100
gcf@npac.syr.edu, furm@npac.syr.edu

Abstract

In our Pragmatic Object Web approach to distributed object technologies, we integrate complementary aspects of CORBA, Java, COM and WOM. Here, we focus on Microsoft COM and we analyze synergies between DirectX / DirectPlay multi-player multimedia gaming technologies and the DoD Modeling and Simulation technologies such as HLA/RTI. We discuss the integration of both layers via the COM/CORBA bridge using our JWORB (Java Web Object Broker) middleware and we outline an early draft of a Web/Commodity based High Performance Virtual Prototyping Environment for Testing, Evaluation and Simulation Based Acquisition that will integrate commodity software (DirectX) and hardware (NT clusters) with the legacy simulation backends.

Introduction

M&S becomes increasingly important for T&E as the fidelity of synthetic environments grows and Virtual Prototyping becomes a realistic cost effective alternative to traditional operational tests. Essential software requirements for Virtual Prototyping Environments to serve T&E needs include: a) integrability of diverse simulation models; b) seemless integration of the simulation engines with the front-end visualization and the back-end datbaase layers; and c) HPC support for simulation engines and parallel I/O operations.

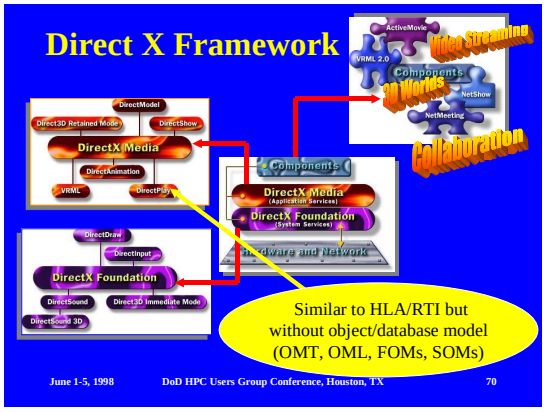

Fig 1: DirectX Framework, organized in three major hierarchy layers: Direct X Foundation or System Services (including all core multimedia support), Direct X Media or Application Services (including DirectPlay) , and high level Components (including NetMeeting, Active Movie and VRML)

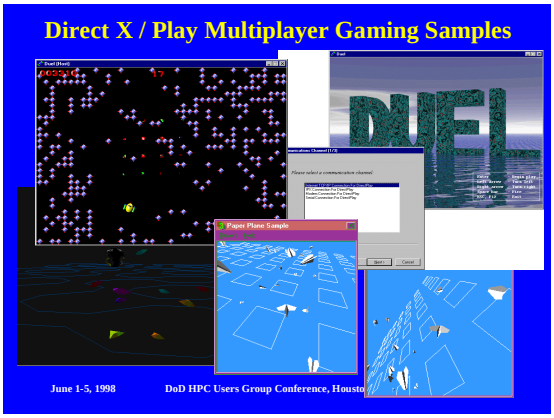

Fig 2: Sample screendumps of Direct X / DirectPlay based networked multi-player games with the real-time 'wargaming' scenarios, distributed as part of the current DirectX releases (v3 for Windows NT, v5 for Windows 95)

Dedicated custom high-end software solutions are hardly competitive today in this area with the new generation Web/Commodity standards such as DCOM, CORBA and Java. At NPAC, we coined the term High Performance Commodity Computing (HPcc) [1] to refer to this new trend

of building new HPC on top of commodity base which we call Pragmatic Object Web [2] and view as an effective integration/interoperability platform for CORBA, Java and DCOM. Our JWORB project [3][4] addresses CORBA/Java integration whereas here we focus on the DCOM based Microsoft technology solution.

Although DCOM is still difficult to use as a low level hacker technology, it is closer than its CORBA and Java competitors to an early but already fully operational prototype solution for a pragmatic HPcc software infrastructure. In the next section, we will review commodity/NT clusters and a family of Microsoft servers, ready to address this new niche for High Performance Computing. We will also discuss there some emergent commodity solutions for other HPC components such as Virtual Interface Architecture (VIA) by Compaq, Intel and Microsoft that addresses scalable parallel I/O between PC clusters.

Microsoft front-ends such as interactive networked multimedia technologies are also getting integrated with COM in the form of the DirectX package that includes 2D graphics, 3D graphics and video/audio multimedia teleconferencing.

An interesting question arises how to integrate optimally the HLA framework that addresses integrability of diverse simulation models with other layers of the Microsoft solution. It turns out that one of the DirectX modules, DirectPlay which supports online gaming and other interactive entertainment applications offers a functionality similar to RTI i.e. a publish/subscribe based event driven support for distributed simulations. When compared with RTI, DirectPlay appears to be a simpler model (for example, it lacks the sophistication of the data Distribution Management or Time Management services of RTI) but it is attractive due to its close integration with other DirectX modules such as DirectSound, DirectDraw, DirectInput or Direct3D.

We are currently conducting research into various aspects of Microsoft HPcc solution outlined above. Our focus is on its potential for the M&S technologies within the FMS and IMT CTAs but we believe that our findings can be of more general relevance for the HPC Modernization Program.

This paper summarizes early results of our research into Microsoft commodity technologies of relevance for DoD Modeling and Simulation. In particular, we describe here: a) our early hands-on experiments with DirectX modules, especially with DirectPlay; b) a more detailed comparison of DirectPlay and RTI software bus technologies for distributed simulations; and c) our integration plan of Microsoft commodity with DoD legacy simulation technologies within the JWORB (Java Web Object Request Broker) based WebHLA framework.

Commodity / NT Clusters as an Emergent HPC Platform

Important new commercial applications, such as internetworking and the WWW, data warehousing and data mining, and enterprise-wide management systems, are driving an explosion in demand for high-end server systems. In the past, commercial users would have looked to mainframes and super-minicomputers - expensive to acquire, maintain, and upgrade - to satisfy these new requirements. But today, servers based on mass-produced personal computer technologies and offering major price / performance advantages over traditional systems are becoming available to meet the needs of the lower end of these new markets.

Clustering commodity PC servers into highly available and adaptable, high-performance computing meshes becomes a viable alternative to the minicomputer or mainframe solutions. Continual increases in the raw power of PC processors and I/O subsystems, combined with the

advent of Microsoft's Windows NT operating system, provide the basic ingredients for building a cluster of PC servers for an enterprise database or other mission critical applications. This type of clusters are commonly referred to as a System Area Network (SAN).

In broad terms, commodity clusters are groups of interconnected servers that work as a single system to provide high-speed, reliable, and scalable service. Until recently, only very expensive, proprietary systems could deliver the levels of speed, reliability, and scalability required for high-performance or enterprise computing. With clustering, less expensive, industry-standard systems now have this capability. Cluster configurations are used to address availability, manageability, and scalability. By Availability, we mean that when a system or application in the cluster fails, the cluster software responds by restarting the failed application or dispersing the work from the failed system to the remaining systems in the cluster. Manageability allows administrators to use a graphical console to move applications and data within the cluster to different servers for load balancing purposes. Finally, we probe Scalability when the overall load for a cluster-aware application exceeds the capabilities of the systems in the cluster and the additional systems can be added transparently to the cluster configuration.

Clustering can take many forms. At the low end, a cluster may be nothing more than a set of standard desktop PCs interconnected by an Ethernet. At the high end, the hardware structure may consist of high performance SMP systems interconnected via a high-performance communications and I/O bus. A client interact with a cluster as though it was a single high-performance, highly reliable server. Additional systems can be added to the clusters as needed to process increasingly complex or numerous requests from clients.

During the evolution of the cluster technology, reliable messaging between clustered servers will become increasingly critical. The VIA initiative is synergistic with current clustering strategies. VIArchitecture was developed to minimize message processing delays and to allow more efficient communication within system area networks. VIArchitecture is the messaging interface for applications that reside on servers connected in a SAN. The concept behind VIA has its roots in the traditional messaging between applications and a computer's Network Interface Controllers. Computer applications operate as if they have unlimited memory. In reality, the OS gives and takes the actual memory away from applications as it is needed to run other applications. Traditionally, if an application wanted to send messages to the NIC using the physical memory, the request had to go through the kernel, which caused processing delays. With VIA, the applicationcan use its virtual memory addresses to speak directly to the SAN NIC without going through the kernel. This will greatly reduce the message latency. Although applications bypass the operating system by sending messages from their virtual addresses, the OS continues to provide security and messaging setup for the applications.

As mentioned previously, VIA was developed by Microsoft, Intel and Compaq. Microsoft is in fact ready to address the full spectrum of new software needs for the cluster computing niche with a famility of server products. This emergent Microsoft solution includes the combination of COM or/and DCOM based software services such as: a) Microsoft Transaction Server (code name Viper) for component management; b) Microsoft Message Queue Server (code name Falcon) for asynchronous messaging; c) Microsoft Cluster Server (code name Wolfpack) for scalable and highly available NT clusters; and d) Microsoft BackOffice Server for core enterprise services (including Exchange Server for Email, NetMeeting and Internet Location Server for Collaboration, SQL Server for Relational Databases etc.).

We are currently starting a new project with Sandia National Laboratory in which we will evaluate these servers, compare them with alternative commodity solutions (Linux, PC Solaris)

and integrate with our JWORB middleware via COM/CORBA bridges. This way, we will construct a high performance two-layer middleware with parallel I/O support between the COM and CORBA software bus architectures. Such framework will enable smooth integration between new Microsoft front-end technologies for real-time multimedia and network multi-player gaming such as DirectX / DirectPlay discussed in the next section and the CORBA/HLA based simulation modules in the backend.

DirectX – Overview

A vital player in transforming Microsoft's Windows Operating System into a major multimedia platform, and in capturing the gaming market, DirectX is a set of APIs for developing powerful graphics, sound and network play applications. The biggest incentive that DirectX offers developers is a consistent interface to devices across different hardware platforms. It achieves this by abstracting away the underlying hardware by using the HAL (Hardware Abstraction Layer) and HEL (Hardware Emulation Layer), both of which are integral parts of the DirectX architecture. This allows for using hardware functionality that might not even be available on the system. DirectX uses fast, low level libraries, which access multimedia hardware in a device independent manner. It thus helps developers get rid of a lot of constraints that influence game design. DirectX, as of version 5, has six categories of APIs to support different device categories and functionality :

- DirectDraw
- Direct3D
- DirectSound
- DirectPlay
- DirectInput
- DirectSetup

Fig 1 collects all elements of the DirectX Framework and it exposes its hierarchical/layered organization, starting from the DirectX Foundation or System Services (including all core multimedia services listed above), followed by DirecrtX Media or Application Services (which includes DirectPlay), and finally followed by a set of high level components such as NetMeeting for collaboration, ActiveMovie for video streaming and so on.

Of these elements, the DirectPlay API is of the most interest to us in our discussion of HLA/RTI. The next section will discuss DirectPlay in a little more detail.

DirectPlay as a part of the DirectX Framework

DirectPlay is a component of DirectX that provides networking and communication services in a transport independent manner. DirectPlay provides a consistent interface for the developer irrespective of whether the transport mechanism is TCP/IP, IPX or modem. This is achieved using 'service providers', which are layered between DirectPlay and the network. A 'service provider' would exist for each transport medium. New ones can be created by 3rd party developers.

A typical DirectPlay gaming scenario would consist of one or more 'sessions'. DirectPlay provides means to create new sessions, list (enumerate) the existing ones and to connect to an

existing session. Once a player application connects to a session, it can interact with other player applications in the session. DirectPlay provides methods to move game data among the various participants. Another interesting feature of DirectPlay is it's Lobby object. The Lobby object has the functionality of a real world lobby where players can meet, interact and find the right partner to play against.

Since the look and feel of the DirectPlay interfaces are similar to those of the DirectX components, with a good application design, DirectPlay can be gracefully merged with other DirectX components to create exciting and stimulating multi-player games.

The DirectPlay API

The DirectPlay API provides the following services for a networked gaming environment : (The functions described below are just used to illustrate the DirectPlay services and are not an exhaustive set of the DirectPlay API functions!)

- *Session Management Functions* A Direct Play session consists of several applications communicating with each other over the network. The session management functions are used to initiate, control and terminate these interactions.

    - EnumSessions(...) : Lists/Enumerates all the sessions in progress on the network. –
    - Open(...) : Creates a new session or connects to an existing one.
    - Close(...) : Disconnects from a session.
    - GetSessionDesc(...) : Obtain the session's current properties.

- *Player Management Functions* The player management funtions are used to manage the players in a session. In addition to creating and destroying players, an application can enumerate the players or retrieve a player's communication capabilities.

    - EnumPlayers(...) : Lists/Enumerates all the players in the session.
    - CreatePlayer(...) : Create a new player.
    - DestroyPlayer(...) : Delete a player.
    - GetPlayerCaps(...) : Get a player's properties (connection speed etc.)

- *Group Management Functions* The group management methods allow applications to create groups of players in a session. Messages can be sent to a group, rather than to one player at a time if the service provider supports multicasting. This is useful to conserve communication-channel bandwidth.

    - EnumGroups(...) : Lists/Enumerates all the groups in the session.

    - CreateGroup(...) : Create a new group.
    - DestroyGroup(...) : Delete a group.
    - EnumGroupPlayers(...) : Lists/Enumerates all the players in a group.
    - AddPlayerToGroup(...) : Adds a new player to a group.
    - DeletePlayerFromGroup(...) : Deletes a player from a group.

- *Message Management Functions* Message management functions are used to route messages among players. Most messages can be defined specific to the application.

- Send(...) : Sends a message to a player, a group, or all the players in the session. –
- Receive(...) : Receives a message from the message queue.
- GetMessageCount(...) : Returns the number of messages wating in the queue.

- *Data Management Functions* DirectPlay lets applications associate data with players and groups. Applications can store two types of information - local and remote. Local data is meant for and is only available to the application that sets it. Remote data is available to all the applications in the session and is used like a shared memory.

  - SetPlayerData(...) : Sets data for a player.
  - GetPlayerData(...) : Gets data for a player.
  - SetGroupData(...) : Sets data for a group.
  - GetGroupData(...) : Gets data for a group.

DirectPlay is based on COM - the Component Object Model and is made up of objects and interfaces based on COM just like the rest of DirectX. All the functions described above are in fact methods of the 'IDirectPlay2' interface of DirectPlay. There are also other DirectPlay library functions used to enumerate the service providers and to create the DirectPlay object itself, before using it's interface.

DirectPlay vs HLA RTI

DirectPlay draws comparisons with the HLA RTI due to the common approach that both take in providing an abstract layer to the network. Although DirectPlay was designed with features specific to gaming, there are several similarities between the HLA RTI and DirectPlay.

A DirectPlay session is analogous to federation execution in HLA, with each application (federate) interacting with one another through a communication medium. Both DirectPlay and RTI provide interfaces and methods to manage sessions/federations. The DirectPlay player management functions are similar to the RTI object management functions for creating and destroying players within a session. The notable difference is that DirectPlay does not provide time synchronization features, and there is no concept of 'declaring' an object's attributes, 'interests' and 'intentions' like in the RTI declaration management service. One of the reasons for this is that a higher framework model doesn't govern DirectPlay as the HLA governs RTI. Another reason is that commercial gaming never required these features. Game developers could decide on their own set of protocols specific to their application.

Next Steps

We are currently in the process of analyzing and testing the most recent DirectX releases (version 3 for Windows NT and version 5 for Windows 95). A collection of screeens with sample games distributed as part of the DirectX Software Development Kit is presented in Fig. 2. Our near term goal is to develop skills in DirectX/DirectPlay game programming and to build suitable bridges between DirectX and HLA/RTI technologies.

A natural framework to enable DirectX/HLA integration is our JWORB based implementation of DMSO RTI. In a simple initial experiment [5] of this type illustrated in Fig. 3, we constructed a 3D front-end version of the Jager video game demo – a DMSO federate distributed as part of the RTI release - and we coupled it with our visual dataflow autoring system WebFlow, operating on

top of the JWORB middlware. Such Web linked interactive real-time simulations will be offered to the M&S community as part of our FMS Training Space under development [6][7][8].

In the pilot effort (Fig. 3), we used OpenGL for graphics to assure UNIX/NT interoperability but we noticed the implied penalty in the graphics performance. In the next step, we are developing a high performance, DirectX / DirectPlay based version of this game. We will also adapt some of the DirectX gaming front-ends of the Jager type such as collected in Fig. 2 so that they can be integrated with or replaced by the RTI based middleware and backend of the Jager game. Such experimentation framework will allow us to explore the DirectPlay / RTI integration issues and to identify the technologies and programming techniques that offer the most adequate high performance commodity solutions.

In the Commodity Clusters project with Sandia National Lab mentioned above, we develop JWORB based high level distributed operating environment that will provide natural linkage between Windows and UNIX software domains. Indeed, JWORB is a multi-protocol server which implements CORBA in Java and connects via COM/CORBA bridge with the COM layers such as DirectX/DirectPlay and via our Object Web (i.e. Java CORBA based) RTI with the DoD Modeling and Simulation domain.

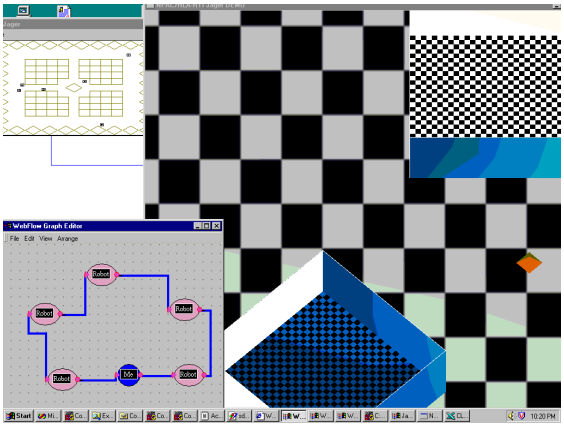

Fig 3: NPAC 3D version of the Jager game, distributed by DMSO as part of the HLA/RTI release. Here we integrate it with WebFlow and Object Model Builder to provide tools for visual authoring and real-time computational steering of WebHLA applications.

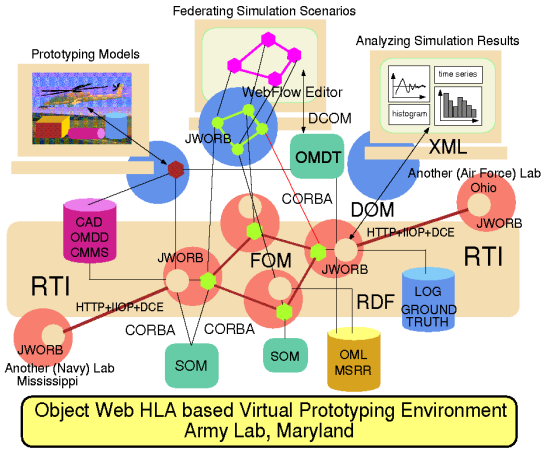

Fig 4: Next step in the WebHLA evolution – a distributed multi-lab Virtual Prototyping Environment for Testing, Evaluation and Simulation based Acquisition which integrates commodity software such as DirectX ahd hardware such as NT clusters with HLA backends.

In consequence, we will be able to construct a poweful next generation M&S environment we call WebHLA that integrates the best features of commodity technologies such as DirectX/DirectPlay and DoD M&S technologies such as HLA/RTI. We view such systems to be of critical relevance to enable Virtual Prototyping Environments for T&E which require high fidelity interactive front-ends combined with high performance legacy simulations and available with commercial quality at affordable price.

Fig. 4 illustrates an overall architecture of such environment which will integrate DirectX for modeling, WebFlow for planning and XML for analysis in the front-end, COM and CORBA in the JWORB / RTI based middleware, and a suite of legacy simulation modules, packaged as COM, CORBA or Enterprise JavaBeans components and ready to plug-and-play on the JWORB / Object RTI software bus.

7. References

1. C G. Fox and W. Furmanski, book chapter in *Building National Grid,* I. Foster and C. Kesselman, editors, Morgan and Kaufman, 1998.

2. C G. Fox, W. Furmanski, H. T. Ozdemir and S. Pallickara, *Building Distributed Systems for the Pragmatic Object Web*, book in progress, Wiley '99.

3. G. C. Fox, W. Furmanski and H. T. Ozdemir, JWORB - Java Web Object Request Broker for Commodity Software based Visual Dataflow Metacomputing Programming Environment , submitted for the HPDC-7, Chicago, IL, July 28-31, 1998.

4. G. C. Fox, W. Furmanski and H. T. Ozdemir, "Object Web (Java/CORBA) based RTI to support Metacomputing M&S", to appear *in Proceedings of the International Test and Evaluation Association (ITEA) Workshop on High Performance Computing for Test and Evaluation*, Aberdeen MD, July 13-16 1998.

5. G. C. Fox, W. Furmanski, B. Goveas, B. Natarajan and S. Shanbhag, "WebFlow based Visual Authoring Tools for HLA Applications", to appear *in Proceedings of the International Test and Evaluation Association (ITEA) Workshop on High Performance Computing for Test and Evaluation,* Aberdeen MD, July 13-16 1998.

6. G. C. Fox, W. Furmanski and T. Pulikal, "Evaluating New Transparent Persistence Commodity Models: JDBC, CORBA PPS and OLEDB for HPC T&E Databases", to appear *in Proceedings of the International Test and Evaluation Association (ITEA) Workshop on High Performance Computing for Test and Evaluation*, Aberdeen MD, July 13-16 1998.

7. D. Bernholdt, G. C. Fox, W. Furmanski, B. Natarajan, H. T. Ozdemir, Z. Odcikin Ozdemir and T. Pulikal, WebHLA - An Interactive Programming and Training Environment for High Performance Modeling and Simulation , *in Proceedings of the DoD HPC 98 Users Group Conference*, Rice University, Houston, TX, June 1-5 1998.

8. G.C.Fox, W. Furmanski, S. Nair, H. T. Ozdemir, Z. Odcikin Ozdemir and T. Pulikal, "WebHLA - An Interactive Programming and Training Environment for High Performance Distributed FMS", to appear *in Proceedings of the Simulation Interoperability Workshop SIW Fall 98*, Orlando, FL, September 14-18, 1998.